WHAT IS CLAIMED IS:

1. A method for load balancing for a plurality of graphics processors configured to operate in parallel, the method comprising:

partitioning a display area into at least a first portion to be rendered by a first one of the plurality of graphics processors and a second portion to be rendered by a second one of the plurality of graphics processors;

instructing the plurality of graphics processors to render a frame, wherein the first and second graphics processors perform rendering for the first and second portions of the display area, respectively;

receiving feedback data for the frame from the first and second graphics processors, the feedback data reflecting respective rendering times for the first and second graphics processors;

determining, based on the feedback data, whether an imbalance exists between respective loads of the first and second graphics processors; and

in the event that an imbalance exists:

identifying, based on the feedback data, which of the first and second graphics processors is more heavily loaded; and

re-partitioning the display area to increase a size of the one of the first and second portions of the display area that is rendered by the more heavily loaded one of the first and second graphics processors and to decrease a size of the other of the first and second portions of the display area.

2. The method of claim 1, wherein the first portion of the display area comprises a first number of contiguous lines of pixels and the second portion of the display area comprises a second number of contiguous lines of pixels.

3. The method of claim 2, wherein the act of re-partitioning the display area includes shifting a third number of contiguous lines of pixels from the first portion of the display area to the second portion of the display area, the third number being smaller than the first number.

4. The method of claim 2, wherein the lines of pixels are oriented horizontally.

5. The method of claim 2, wherein the lines of pixels are oriented vertically.

6. The method of claim 1, further comprising:

assigning a different processor identifier to each of the first and second graphics processors,

wherein the feedback data received from each of the first and second graphics processors includes the respective processor identifier.

7. The method of claim 6, wherein each of the processor identifiers has a numerical value.

8. The method of claim 1, wherein the feedback data includes a timestamp.

9. The method of claim 1, wherein the feedback data includes data indicating which of the first and second graphics processors is last to finish rendering the frame.

10. The method of claim 9, wherein the feedback data from the one of the first and second graphics processors that is last to finish overwrites feedback data from the other of the first and second graphics processors.

11. The method of claim 9, wherein the act of receiving includes receiving the feedback data for each of a plurality of frames.

12. The method of claim 11, further comprising:

providing a plurality of storage locations, each storage location associated with a different one of the plurality of frames,

wherein the act of receiving the feedback data for each of the plurality of frames includes:

instructing the first graphics processor to store a first processor identifier in the associated one of the storage locations for each of the plurality of frames after rendering the first portion of the display area for the frame; and

instructing the second graphics processor to store a second processor identifier different from the first processor identifier in the associated one of the storage locations for each of the plurality of frames after rendering the second portion of the display area for the frame.

13. The method of claim 12, wherein the processor identifier of the one of the of the first and second graphics processors that was last to finish rendering the frame overwrites the processor identifier of the other of the first and second graphics processors in the storage location.

14. The method of claim 11, wherein the act of determining includes computing a load coefficient from the feedback data for the plurality of frames, the load coefficient indicating a frequency of one of the first and second graphics processors being last to finish.

15. The method of claim 14, wherein a numeric identifier is associated with each of the first and second graphics processors and the load coefficient is an average over the plurality of frames of the numeric identifier of the processor that was last to finish each frame.

16. The method of claim 15, wherein the act of determining further includes comparing the load coefficient to an arithmetic mean of the numeric identifiers.

17. The method of claim 15, wherein during the act of re-partitioning, an amount by which the size of the first portion of the display area is reduced depends at least in part on a magnitude of the difference between the load coefficient and the arithmetic mean.

18. The method of claim 1, further comprising:

generating a command stream for each of the first and second graphics processors, the command stream including a set of rendering commands for the frame; and

inserting a write notifier command into a command stream for each of the first and second graphics processors following the set of rendering commands, wherein each of the first and second graphics processors responds to the write notifier command by transmitting the feedback data to a storage location.

19. A graphics processing system comprising:

a graphics driver module; and

a plurality of graphics processors configured to operate in parallel to render respective portions of a display area and to provide feedback data to the graphics driver module,

the graphics driver module being further configured to detect, based on the feedback data, an imbalance between respective loads of two of the plurality of graphics processors and, in response to detecting an imbalance, to decrease a size of a first portion of the display area that is rendered by a more heavily loaded one of the two graphics processors and to increase a size of a second portion of the display area that is rendered by the other one of the two graphics processors.

20. The graphics processing system of claim 19, further comprising a plurality of graphics memories, each graphics memory coupled to a respective one of the graphics processors and storing pixel data for the portion of the display area rendered by the graphics processor coupled thereto.

21. The graphics processing system of claim 20, further comprising scanout control logic coupled to the plurality of graphics memories and configured to read pixel data for the display area from the graphics memories.

22. The graphics processing system of claim 19, wherein the graphics driver module is further configured to generate a command stream for the plurality of graphics processors, the command stream including a set of rendering commands for a frame and an instruction to each of the two graphics processors to transmit feedback data indicating that the transmitting processor has executed the set of rendering commands.

23. The graphics processing system of claim 19, wherein the feedback data includes an indication of which of the two graphics processors was last to finish rendering a frame.

24. The graphics processing system of claim 23, wherein the feedback data includes a numeric identifier of the one of the two graphics processors that was last to finish and the graphics driver module is further configured to compute a load coefficient from the numeric identifiers over a plurality of frames.

25. The graphics processing system of claim 24, wherein the graphics driver module is further configured to detect an imbalance in the event that the load coefficient is greater than a high threshold or less than a low threshold.

26. The graphics processing system of claim 19, wherein each portion of the display area comprises a number of contiguous lines of pixels and wherein the two graphics processors are configured to render adjacent portions.

27. The graphics processing system of claim 26, wherein the graphics driver is further configured to decrease the size of the first portion and increase the size of the second portion by shifting a number of lines of pixels from the first portion to the second portion.